NASA TM-84605

NASA-TM-84605 19830012318

NASA Technical Memorandum 84605

NETWORK SIMULATION USING THE SIMULATION LANGUAGE FOR ALTERNATIVE MODELING (SLAM II)

STEWART SHEN
W. DOUGLAS MORRIS

FEBRUARY 1983

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

NETWORK SIMULATION USING

THE SIMULATION LANGUAGE FOR ALTERNATE MODELING (SLAM II)

by Stewart N. T. Shen* and W. Douglas Morris

Abstract

Computer simulation is a widely used technique for problem solving, system analysis, and design. Many specialized languages have been developed as an aid to implementing simulation studies. The simulation language for alternative modeling (SLAM II) is a general purpose language that combines network, discrete event, and continuous modeling capabilities in a single language system. The efficacy of the system's network modeling is examined and discussed. Examples are given of the symbolism that is used, and an example problem and model are derived. The results are discussed in terms of the ease of programming, special features, and system limitations. The system offers many features which allow rapid model development and provides an informative standardized output. The system also has limitations which may cause undetected errors and misleading reports unless the user is aware of these programming characteristics.

I. Introduction

SLAM is a FORTRAN based general purpose simulation language that was introduced in 1979 by Pritsker [1]. The language was later enhanced and called SLAM II [2].

SLAM II provides the network or block type of simulation capability similar to GPSS [3], thus it allows convenient implementation of many types of simulation problems. By providing the capability for discrete event simulation as in SIMSCRIPT [4] and SIMULA [5], this allows the user to incorporate computations and logic flows into the simulation that would not be possible using

*1982 Summer NASA/ASEE Faculty Fellow.

network simulation alone. It allows direct calling from the network model to FORTRAN subroutines and functions under certain situations. Entities, which correspond to the transactions in GPSS, can be entered into the network by event routines. This allows the modification in FORTRAN subroutines of the network status such as the GATE and the RESOURCE, which is similar to the FACILITY or STORAGE in the General Purpose System Simulation (GPSS) language. Thus, at least conceptually, SLAM II has the power of an event type of simulation language and the convenience of a block or network type of simulation language. In addition, it allows the flexibility of combining the two different views in a single simulation.

SLAM II also provides continuous simulation capabilities. Continuous simulation may be used in conjunction with discrete simulation in a single model. This again is an excellent feature.

SLAM II is a relatively new simulation language and has a small user population, but its popularity is growing and is quite significant.

The discrete event simulation and the continuous simulation capabilities of SLAM II are similar to other languages but offer the advantage of being used in conjunction with the network simulation. This paper will concentrate on the discrete simulation aspect of SLAM II (Version 4, release 3). In particular, the network modeling capabilities will be reviewed and discussed.

II. Network Modeling with SLAM II

A SLAM network model is a network representation of a process through which entities flow. Entities are abstract representations of persons or things. They may be described with attributes. For example, automobiles may be an entity type with three attributes to describe them: make, year, and model. A SLAM network consists of nodes and branches. The nodes are the actions to be taken.

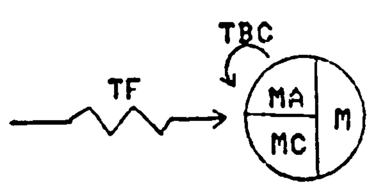
Examples are the CREATE node which generates entities; the QUEUE node where an entity tries to enter a queue; and the ASSIGN node where an entity's passing through will cause values to be assigned to attributes or variables. The branches are the representation of the times taken for the activities performed on the entities or by the entities before reaching other nodes. The time spent between seizing and releasing a resource, for instance, is represented by a branch in the network.

As in GPSS, many output reports are either automatic or optional. Hence, required programming efforts are typically minimal. On the other hand, elaborate output reports may be custom designed since FORTRAN subroutines and functions can be called from the network.

1. Symbolisms in SLAM II network modeling

Graphic symbols are used to represent the nodes and the branches of the network. It is recommended that a simulation model be graphically built first and then be coded in the input statements. Some examples of the symbols and their corresponding network input statements are given below for the purpose of illustration.

(1) The CREATE node:



The CREATE statement:

CREATE, TBC, TF, MA, MC, M;

The network input statements all start in column 7 or later with a key word followed by parameters separated by commas and end with a semicolon. The CREATE node generates entities according to the given parameter specification. The meanings of the parameters of CREATE are given below.

TBC: The interval between succeeding entity generations. It may be a constant, a SLAM variable to be defined for each simulation, or a SLAM random variable whose value to be determined by the system. The default value is ∞.

TF: The time of the first entity generation. It must be a constant. Its default value is 0.

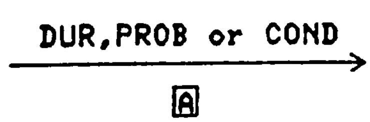
MA: The time of each generation is marked in the MA^{th} attribute of the corresponding entity. It must be a positive integer constant. If this parameter is missing, the generation time is not marked.

MC: The maximum number of entities to be generated from this node. Its default value is ∞.

M: The maximum number of activities to be initiated. The default value is ∞. Note that there may be several activities, or branches, given after a node. If M is greater than 1, then the entity leaving this node may split into several to travel through the several appropriate activities independently.

All the parameters of SLAM input statements are position significant. For example, the third parameter for CREATE would always be interpreted as MA.

(2) The Regular Activity branch:



The Regular Activity statement:

ACTIVITY/A, DUR, PROB or COND, NLBL;

The Regular Activity is used to perform probablistic or conditional testing, to delay for a specified duration, and to route entities to non-sequential nodes. The parameters are explained below.

A: Index of the activity. It must be between 1 and 50, inclusive. Its default is no indexing. Only indexed activities will have statistics maintained and reported. (The upper limit of 50 may be a serious limitation on a large simulation study.)

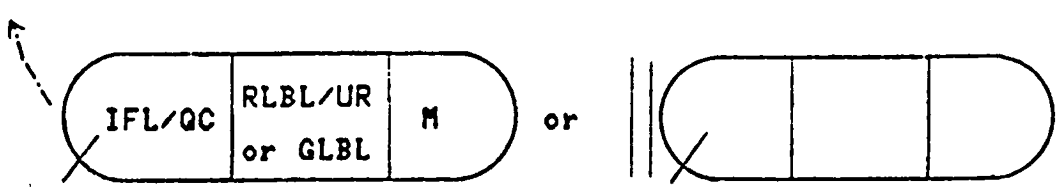
DUR: The duration of the activity. It may be a constant, a SLAM variable, a SLAM random variable, the next release or generation time of a given CREATE node, a special stopping time to be set in an event routine, or a user function call.

PROB: A probability specification, a constant between 0 and 1. The sum of all probabilities of the activities with probabilities eminating from a node should equal to 1.

COND: A logical expression on attributes, SLAM variables, SLAM random variables, or constants. Its default value is true.

NLBL: The label of a node where the entity may be routed. Lack of a label causes the entity to be routed to the next node.

(3) The AWAIT node:



The AWAIT statement:

AWAIT (IFL/QC), RLBL/UR or GLBL, BLOCK or BALK (NLBL), M;

The AWAIT node stores entities on a file until the corresponding gate opens or the corresponding resource has enough units for serving one entity on the file. In the later case, an entity will leave the node only when sufficient units of the resource become available. These units of resource are considered seized immediately and become temporarily unavailable to other entities.

The meanings of the parameters are given below.

IFL: The file number of the file to store the entities. It must be a positive integer constant less than or equal to the maximum number of files declared on the LIMITS card. (The LIMITS card will be illustrated later.)

QC: A constant representing the capacity of the AWAIT node. When this number of entities are stored in file IFL, no more entities may enter the AWAIT node until after some entities have left.

RLBL/UR or GLBL: The corresponding resource RLBL or gate GLBL. In case of resource, UR is the number of units of the resource required by the entity.

BLOCK or BALK (NLBL): If an entity cannot enter the AWAIT node, then the entity is BLOCKed, thus holding up the resource activity it comes from; the entity BALKs and leaves for the node with label NLBL; or the entity is destroyed from the system if neither BLOCK nor BALK is specified. Note that BALK and BLOCK are key words.

M: The maximum number of activities which the entity may split up and then pass through. It is the same as the M in CREATE.

2. Sample Simulation Problem and Model

A sample simulation model may best illustrate the SLAM II network concept. Consider the following hypothetical problem.

Space shuttle orbiters are delivered to a service hanger every 30 time units and a total of five are to be delivered. The service hanger has a service capacity of two orbiters simultaneously. The processing time at the service hanger is in normal distribution with mean of five time units and standard deviation of one time unit. There is one orbiter transporter available to move an orbiter from the hanger to the launch pad. The transportation time is 0.5 time unit. The orbiter transporter becomes available at the service hanger again in one time unit after the orbiter has arrived at the launch pad and the launch pad is available for mounting the orbiter. Further processing on the orbiter at the launch pad is one time unit for the first simulation and 1.5 time units for the second simulation. The Shuttle has to wait for a launch window to open before it can be launched. The window opens for 0.5 time unit then closes for one time, unit. This situation repeats continuously. The launch takes a negligible amount of time and the travel time in space is in exponential distribution with mean of four time units in the first simulation and five time units in the second simulation. The landing takes a negligible time, and the orbiter is sent right back to the service hanger for reprocessing. This whole operation is to be examined for a period of 500 time units. Do the simulation twice with the changes mentioned above. A schematic of the system is shown in Figure 1.

The graphic representation of the SLAM II simulation model consists of two disjoint networks as given in Figures 2 and 3.

The SLAM II model in input statements is given in Figure 4. Lines 4 through 32 are the network input statements. Comments are given on the right-hand side of the listing. Some further explanations are given in Section 3.

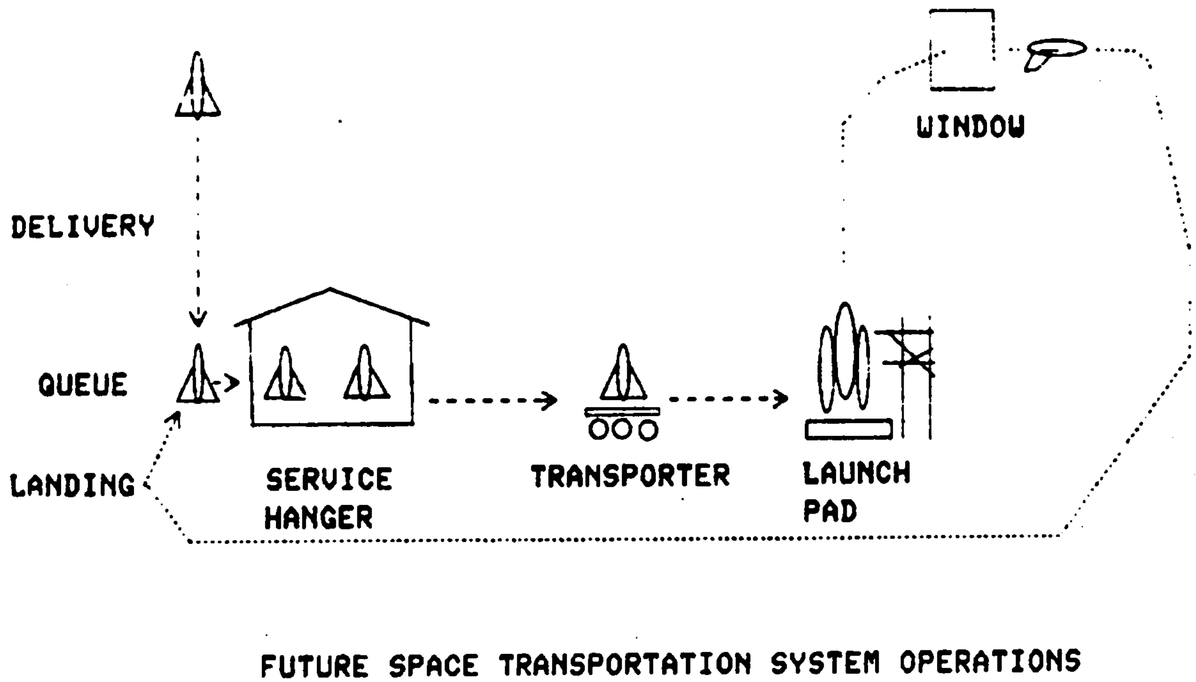


Figure 1 - Schematic of hypothetical system.

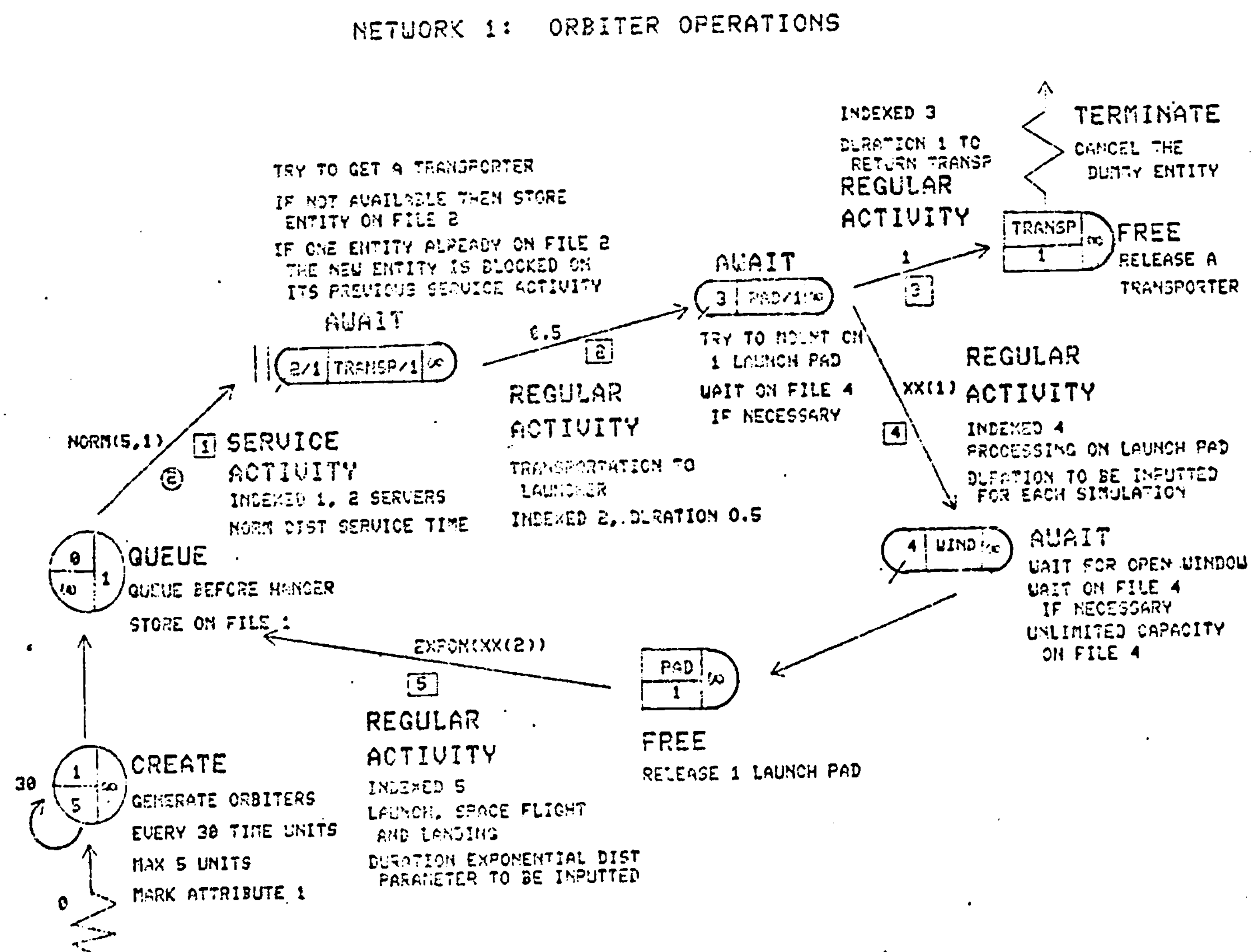


Figure 2 - Orbiter operation network.

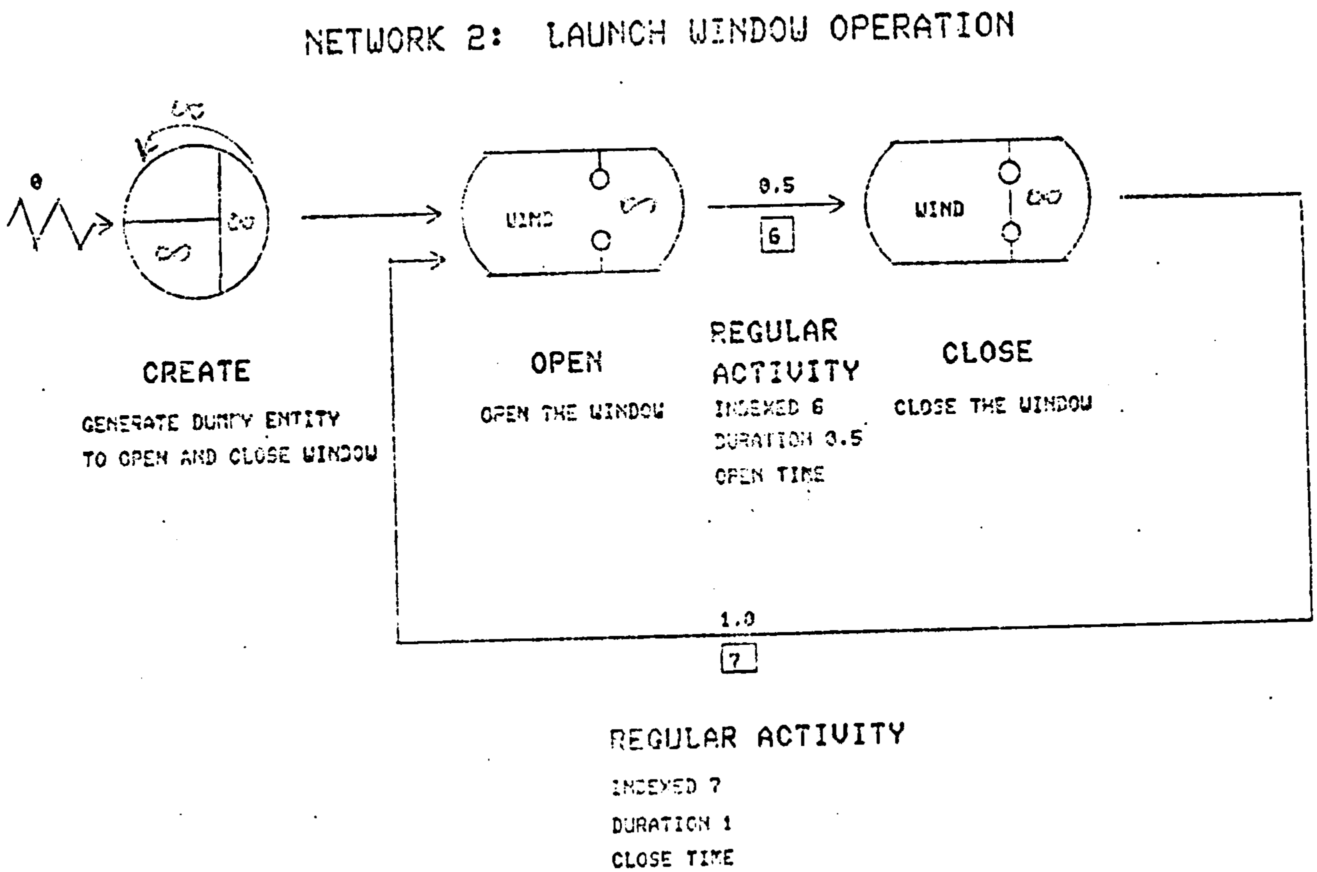


Figure 3 - Launch window network.

```
GEN,SHEN & MORRIS,HYPOTHETICAL SSS4,6/18/82,2;
LIMITS,4,2,10;
NETWORK;
       RESOURCE/TRANSP(1),2;        ONE ORBITER TRANSP,WAITINGS ON FILE 2
       RESOURCE/PAD(1),3;           ONE LAUNCH PAD, WAITINGS ON FILE 3
       GATE/WIND,OPEN,4;            WINDOW IS A GATE, INITIALLY OPEN,
;                                    WAITINGS ON FILE 4
LA1    CREATE,30,0,1,5;             CREATES ORBITERS EVERY 30 TIME UNITS,
;                                    MAXIMUN 5 UNITS, MARK ATTRIBUTE 1
HNGQ   QUEUE(1);                    ENTER QUEUE FOR HANGER,WAIT ON FILE 1
         ACT(2)/1,RNORM(5.,1.);     REPROCESS ORBITER, CAP 2 SERVERS,
;                                    DURATION IS NORMAL DISTRIBUTION
LA2    AWAIT(2/1),TRANSP/1,BLOCK;   WAIT ON FILE 2 FOR ONE TRANSP, CAP 1,
;                                    BLOCK NEW ENTITY ON PREVIOUS ACTIVIT
;                                    IF CAPACITY ALREADY REACHED
         ACT/2,0.5;                 MOVE TO LUNCH PAD, 0.5 TIME UNIT
LA3    AWAIT(3),PAD/1;              WAIT ON FILE 3 FOR ONE PAD
         ACT/3,1,,FREM;             ONE TIME UNIT TO RETURN TRANSP
         ACT/4,XX(1),,WATWN;        XX(1) TIME UNITS TO PROCESS ON PAD,
;                                    WAIT FOR WINDOW
FREM   FREE,TRANSP/1;               RETURNS TRANSP TO HANGER
LA4    TERMINATE;                   DESTROY DUMMY ENTITY
WATWN  AWAIT(4),WIND;               WAIT FOR WINDOW TO OPEN
LA5    FREE,PAD/1;                  PAD BECOMES AVAILABLE
         ACT/5,EXPON(XX(2)),,HNGQ;  LAUNCH, TRAVEL IN SPACE, AND RETURN
;                                    DURATION EXPONENTIAL TIME
LA6    CREATE;                      GENERATES DUMMY ENTITY TO OPEN AND
;                                    CLOSE THE WEATHER WINDOW
OPWN   OPEN,WIND;                   WINDOW OPENED
         ACT/6,0.5;                 LENGTH OF WINDOW OPEN
LA7    CLOSE,WIND;                  WINDOW CLOSED
         ACT/7,1.,,OPWN;            LENGTH OF WINDOW CLOSED, BACK TO OPEN
       ENDNETWORK;
INITIALIZE,0,500;
MONTR,TRACE,0,100,1;
INTLC,XX(1)=1.,XX(2)=4.;
SIMULATE;
MONTR,TRACE,0,100,1;
INTLC,XX(1)=1.5,XX(2)=5.;
FIN;
```

Figure 4 - SLAM II model of the hypothetical system.

3. <u>The Control Statements</u>

To execute the SLAM II network model requires some additional control statements. Lines 1, 2, and 3, and lines 33 through 40 are such statements. Line 1, or the GEN statement, gives the author, project name, the date, and the number of simulations. Line 2, the LIMITS statement, gives the maximum number of files used, the largest number of attributes per entity, and the maximum number of concurrent entities in all files. Line 3, the NETWORK statement, indicates the start of the network statements. Line 33, the END NETWORK statement, indicates the end of the network statements. Line 34, the INITIALIZE statement, gives the beginning and the ending times of the simulation. Line 35, the MONTR statement, requests a trace of the simulation from time 0 through time 100 with the first attribute of each entity (which is the time when the entity was created) printed in the trace. Line 36, the INTLC statement, initializes the SLAM variables for the simulation run. Line 37, the SIMULATE statement, instructs the SLAM II system to perform a simulation run with the information given so far. Lines 38 and 39 are for the next simulation run. Line 40, the FIN statement, is equivalent to SIMULATE except it is required for the last simulation run. Some of the control statements allow additional parameters for other possibilities, which have not been described.

4. <u>Simulation output</u>

Two separate simulation runs were made, but only the output of the first run is given in Figure 5. To further conserve space, only a partial list of the trace report is given. It is recommended that a reader with no previous experience in network or block type of simulation language should follow the trace carefully to see how entities flow in the network.

The output is quite self-explanatory. In the trace, TNOW is the clock time. The first attribute of each entity is displayed in the trace. It is the entity creation time and can be used to help identify an entity.

```
                         S L A M   E C H O   R E P O R T

              SIMULATION PROJECT HYPOTHETICAL SSS4          BY SHEN & MORRIS

              DATE  6/18/1982                               RUN NUMBER    1 OF    2

                              SLAM VERSION JAN 79

GENERAL OPTIONS

      PRINT INPUT STATEMENTS (ILIST):                YES
      PRINT ECHO REPORT (IECHO):                     YES
      EXECUTE SIMULATIONS (IXQT):                    YES
      PRINT INTERMEDIATE RESULTS HEADING (IPIRH):    YES
      PRINT SUMMARY REPORT (ISMRY):                  YES

LIMITS ON FILES

      MAXIMUM NUMBER OF USER FILES (MFILS):                 4
      MAXIMUM NUMBER OF USER ATTRIBUTES (MATR):             2
      MAXIMUM NUMBER OF CONCURRENT ENTRIES (MNTRY):        10

FILE SUMMARY

      FILE           INITIAL       RANKING
      NUMBER         ENTRIES       CRITERION

         1              0            FIFO
         2              0            FIFO
         3              0            FIFO
         4              0            FIFO

RANDOM NUMBER STREAMS

      STREAM         SEED         REINITIALIZATION
      NUMBER         VALUE        OF STREAM

         1  124397822910357       NO
         2    3467133363289       NO
         3   75654468614391       NO
         4  184172232130623       NO
         5  280033229735385       NO
         6  147959512939949       NO
         7  125894553354929       NO
         8  150477775683725       NO
         9  227874749727917       NO
        10   82174077946221       NO

INITIALIZATION OPTIONS

      BEGINNING TIME OF SIMULATION (TTBEG):  0.
      ENDING TIME OF SIMULATION (TTFIN):      .5000E+03
      STATISTICAL ARRAYS CLEARED (JJCLR):     YES
      VARIABLES INITIALIZED (JJVAR):          YES
      FILES INITIALIZED (JJFIL):              YES
```

Figure 5 - Simulation output.

NSET/QSET STORAGE ALLOCATION

DIMENSION OF NSET/QSET (NNSET):	5000
WORDS ALLOCATED TO FILING SYSTEM:	60
WORDS ALLOCATED TO INDEXED LIST TAGS:	0
WORDS ALLOCATED TO NETWORK:	238
WORDS AVAILABLE FOR PLOTS/TABLES:	4702

INPUT ERRORS DETECTED: 0

EXECUTION WILL BE ATTEMPTED

1 **INTERMEDIATE RESULTS**

1

SLAM TRACE BEGINNING AT TNOW= 0.

TNOW	JEVNT	NODE ARRIVAL LABEL	NODE ARRIVAL TYPE	ATTRIBUTE : 1	CURRENT ATTRIBUTE BUFFER	REGULAR ACTIVITY SUMMARY INDEX	DURATION	END NODE
0.		LA1	CREATE	0.				
0.		HNGQ	QUEUE	0.				
0.		LA6	CREATE	0.				
0.		OPUN	OPEN	0.		6	.5000	LA7
.5000E+00		LA7	CLOSE	0.		7	1.0000	OPUN
.1500E+01		OPUN	OPEN	0.		6	.5000	LA7
.2000E+01		LA7	CLOSE	0.		7	1.0000	OPUN
.2650E+01		LA2	AWAIT	0.		2	.5000	LA3
.3000E+01		OPUN	OPEN	0.		6	.5000	LA7
.3150E+01		LA3	AWAIT	0.		3	1.0000	FREM
						4	1.0000	WATU
.3500E+01		LA7	CLOSE	0.		7	1.0000	OPUN
.4150E+01		FREM	FREE	0.				
.4150E+01		LA4	TERM	0.				
.4150E+01		WATU	AWAIT	0.				
.4500E+01		OPUN	OPEN	0.		6	.5000	LA7
.4520E+01		LA5	FREE	0.		5	5.9630	HNGQ
.5000E+01		LA7	CLOSE	0.		7	1.0000	OPUN
.6000E+01		OPUN	OPEN	0.		6	.5000	LA7
.6500E+01		LA7	CLOSE	0.				

Figure 5 (Continued)

			7	1.0000	OPUN
.7500E+01	OPUN OPEN	0.	6	.5000	LA7
.8000E+01	LA7 CLOSE	0.	7	1.0000	OPUN
.9000E+01	OPUN OPEN	0.	6	.5000	LA7
.9500E+01	LA7 CLOSE	0.	7	1.0000	OPUN
.1046E+02	HNGO QUEUE	0.			
.1050E+02	OPUN OPEN	0.	6	.5000	LA7
.1100E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.1200E+02	OPUN OPEN	0.	6	.5000	LA7
.1250E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.1350E+02	OPUN OPEN	0.	6	.5000	LA7
.1400E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.1520E+02	OPUN OPEN	0.	6	.5000	LA7
.1550E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.1650E+02	OPUN OPEN	0.	6	.5000	LA7
.1700E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.1713E+02	LA2 AWAIT	0.	2	.5000	LA3
.1763E+02	LA3 AWAIT	0.	3	1.0000	FREM
			4	1.0000	WATW
.1800E+02	OPUN OPEN	0.	6	.5000	LA7
.1850E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.1863E+02	FREM FREE	0.			
.1863E+02	LA4 TERM	0.			
.1863E+02	WATW AWAIT	0.			
.1950E+02	OPUN OPEN	0.	6	.5000	LA7
.1950E+02	LA5 FREE	0.	5	.5188	HNGO
.2000E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.2000E+02	HNGO QUEUE	0.			
.2100E+02	OPUN OPEN	0.	6	.5000	LA7
.2150E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.2250E+02	OPUN OPEN	0.	6	.5000	LA7
.2300E+02	LA7 CLOSE	0.	7	1.0000	OPUN
.2400E+02	OPUN OPEN	0.	6	.5000	LA7
.2433E+02	LA2 AWAIT	0.	2	.5000	LA3
.2450E+02	LA7 CLOSE	0.			

Figure 5 (Continued)

.2483E+02	LA3	AWAIT	0.
.2550E+02	OPUN	OPEN	0.
.2583E+02	FREM	FREE	0.
.2583E+02	LA4	TERM	0.
.2583E+02	WATU	AWAIT	0.
.2583E+02	LA5	FREE	0.
.2600E+02	LA7	CLOSE	0.
.2690E+02	HNGQ	QUEUE	0.
.2700E+02	OPUN	OPEN	0.
.2750E+02	LA7	CLOSE	0.
.2850E+02	OPUN	OPEN	0.
.2900E+02	LA7	CLOSE	0.
.3000E+02	LA1	CREATE	.3000E+02
.3000E+02	HNGQ	QUEUE	.3000E+02
.3000E+02	OPUN	OPEN	0.
.3050E+02	LA7	CLOSE	0.
.3150E+02	OPUN	OPEN	0.
.3200E+02	LA7	CLOSE	0.
.3300E+02	OPUN	OPEN	0.
.3350E+02	LA7	CLOSE	0.
.3355E+02	LA2	AWAIT	0.
.3445E+02	LA3	AWAIT	0.
.3450E+02	OPUN	OPEN	0.
.3500E+02	LA7	CLOSE	0.
.3520E+02	LA2	AWAIT	.3000E+02
.3545E+02	FREM	FREE	0.
.3545E+02	LA4	TERM	0.
.3545E+02	WATU	AWAIT	0.
.3595E+02	LA3	AWAIT	.3000E+02
.3600E+02	OPUN	OPEN	0.
.3600E+02	LA5	FREE	0.
.3650E+02	LA7	CLOSE	0.
.3700E+02	FREM	FREE	.3000E+02
.3700E+02	LA4	TERM	.3000E+02
.3700E+02	WATU	AWAIT	.3000E+02

7	1.0000	OPUN
3	1.0000	FREM
4	1.0000	WATU
6	.5000	LA7
5	1.0635	HNGQ
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
2	.5000	LA3
3	1.0000	FREM
4	1.0000	WATU
6	.5000	LA7
7	1.0000	OPUN
2	.5000	LA3
6	.5000	LA7
3	1.0000	FREM
4	1.0000	WATU
5	8.4389	HNGQ
7	1.0000	OPUN

Figure 5 (Continued)

.3750E+02	OPUN	OPEN	0.
.3750E+02	LA5	FREE	.3000E+02
.3800E+02	LA7	CLOSE	0.
.3900E+02	OPUN	OPEN	0.
.3950E+02	LA7	CLOSE	0.
.4050E+02	OPUN	OPEN	0.
.4100E+02	LA7	CLOSE	0.
.4200E+02	OPUN	OPEN	0.
.4250E+02	LA7	CLOSE	0.
.4350E+02	OPUN	OPEN	0.
.4400E+02	LA7	CLOSE	0.
.4444E+02	HNGQ	CLEJE	0.
.4520E+02	OPUN	OPEN	0.
.4550E+02	LA7	CLOSE	0.
.4593E+02	HNGQ	CLEJE	.3000E+02
.4650E+02	OPUN	OPEN	0.
.4700E+02	LA7	CLOSE	0.
.4800E+02	OPUN	OPEN	0.
.4850E+02	LA7	CLOSE	0.
.4950E+02	OPUN	OPEN	0.
.5000E+02	LA7	CLOSE	0.
.5068E+02	LA2	AWAIT	.3000E+02
.5097E+02	LA2	AWAIT	0.
.5100E+02	OPUN	OPEN	0.
.5118E+02	LA3	AWAIT	.3000E+02
.5150E+02	LA7	CLOSE	0.
.5218E+02	FREM	FREE	.3000E+02
.5218E+02	LA4	TERM	.3000E+02
.5218E+02	WATU	AWAIT	.3000E+02
.5250E+02	OPUN	OPEN	0.
.5250E+02	LA5	FREE	.3000E+02
.5280E+02	LA3	AWAIT	0.
.5300E+02	LA7	CLOSE	0.

6	.5000	LA7
5	8.4201	HNGQ
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
6	.5000	LA7
7	1.0000	OPUN
2	.5000	LA3
6	.5000	LA7
3	1.0000	FREM
4	1.0000	WATU
7	1.0000	OPUN
2	.5000	LA3
6	.5000	LA7
5	1.2807	HNGQ
3	1.0000	FREM
4	1.0000	WATU

Figure 5 (Continued)

				7	1.0000	OPUN
.5369E+02	FREM	FREE	0.			
.5369E+02	LA4	TERM	0.			
.5369E+02	WATU	AWAIT	0.			
.5376E+02	HNGQ	QUEUE	.3000E+02			
.5400E+02	OPUN	OPEN	0.	6	.5000	LA7
.5400E+02	LA5	FREE	0.	5	4.0741	HNGQ
.5450E+02	LA7	CLOSE	0.	7	1.0000	OPUN
.5550E+02	OPUN	OPEN	0.	6	.5000	LA7
.5600E+02	LA7	CLOSE	0.	7	1.0000	OPUN
.5700E+02	OPUN	OPEN	0.	6	.5000	LA7
.5750E+02	LA7	CLOSE	0.	7	1.0000	OPUN
.5807E+02	HNGQ	QUEUE	0.			
.5850E+02	OPUN	OPEN	0.	6	.5000	LA7
.5900E+02	LA7	CLOSE	0.	7	1.0000	OPUN
.6000E+02	LA1	CREATE	.6002E+02			
.6000E+02	HNGQ	QUEUE	.6032E+02			
.6000E+02	OPUN	OPEN	0.	6	.5000	LA7
.6024E+02	LA2	AWAIT	.3000E+02	2	.5000	LA3
.6050E+02	LA7	CLOSE	0.	7	1.0000	OPUN
.6074E+02	LA3	AWAIT	.3000E+02	3	1.0000	FREM
				4	1.0000	WATU
.6150E+02	OPUN	OPEN	0.	6	.5000	LA7
.6174E+02	FREM	FREE	.3000E+02			
.6174E+02	LA4	TERM	.3000E+02			
.6174E+02	WATU	AWAIT	.3000E+02			
.6174E+02	LA5	FREE	.3000E+02	5	.5876	HNGQ
.6196E+02	LA2	AWAIT	0.	2	.5000	LA3
.6220E+02	LA7	CLOSE	0.	7	1.0000	OPUN
.6233E+02	HNGQ	QUEUE	.3000E+02			
.6246E+02	LA3	AWAIT	0.	3	1.0000	FREM
				4	1.0000	WATU
.6300E+02	OPUN	OPEN	0.	6	.5000	LA7
.6346E+02	FREM	FREE	0.			
.6346E+02	LA4	TERM	0.			
.6346E+02	WATU	AWAIT	0.			
.6346E+02	LA5	FREE	0.	5	5.3879	HNGQ
.6350E+02	LA7	CLOSE	0.	7	1.0000	OPUN
.6450E+02	OPUN	OPEN	0.	6	.5000	LA7
.6500E+02	LA7	CLOSE	0.			

Figure 5 (Continued)

S L A M   S U M M A R Y   R E P O R T

SIMULATION PROJECT HYPOTHETICAL SSS4   BY SHEN & MORRIS

DATE 6/18/1982   RUN NUMBER 1 OF 2

CURRENT TIME .5000E+03
STATISTICAL ARRAYS CLEARED AT TIME 0.

FILE STATISTICS

FILE NUMBER	ASSOCIATED NODE TYPE	AVERAGE LENGTH	STANDARD DEVIATION	MAXIMUM LENGTH	CURRENT LENGTH	AVERAGE WAITING TIME
1	QUEUE	.7069	.8218	3	0	2.0431
2	AWAIT	.0764	.2656	1	0	.2234
3	AWAIT	.0104	.1014	1	0	.0304
4	AWAIT	.1158	.3200	1	0	.3406
5		5.2715	1.0898	8	7	.9427

REGULAR ACTIVITY STATISTICS

ACTIVITY INDEX	AVERAGE UTILIZATION	STANDARD DEVIATION	MAXIMUM UTILIZATION	CURRENT UTILIZATION	ENTITY COUNT
2	.1710	.3765	1	0	171
3	.3410	.4740	1	1	170
4	.3410	.4740	1	1	170
5	1.2639	.9132	4	2	162
6	.3340	.4716	1	0	334
7	.6660	.4716	1	1	333

SERVICE ACTIVITY STATISTICS

ACTIVITY INDEX	START NODE LABEL/TYPE	SERVER CAPACITY	AVERAGE UTILIZATION	STANDARD DEVIATION	CURRENT UTILIZATION	AVERAGE BLOCKAGE	MAXIMUM IDLE TIME/SERVERS	MAXIMUM BUSY TIME/SERVERS	ENTITY COUNT
1	HNGO QUEUE	2	1.7146	.5782	2	0.0000	2.0000	2.0000	171

RESOURCE STATISTICS

RESOURCE NUMBER	RESOURCE LABEL	CURRENT CAPACITY	AVERAGE UTILIZATION	STANDARD DEVIATION	MAXIMUM UTILIZATION	CURRENT UTILIZATION
1	TRANSP	1	.5224	.4995	1	1
2	PAD	1	.4568	.4981	1	1

RESOURCE NUMBER	RESOURCE LABEL	CURRENT AVAILABLE	AVERAGE AVAILABLE	MINIMUM AVAILABLE	MAXIMUM AVAILABLE
1	TRANSP	0	.4776	0	1
2	PAD	0	.5432	0	1

GATE STATISTICS

GATE NUMBER	GATE LABEL	CURRENT STATUS	PCT. OF TIME OPEN
1	WIND	CLOSED	.3340

Figure 5 (Concluded)

III. Evaluation of the SLAM II Network Simulation Features

Conceptually, SLAM II network simulation features are an improvement over the languages such as GPSS. The hypothetical model given earlier illustrates the convenient implementation of a simulation model using SLAM II. In addition, SLAM II allows arbitrarily complex operations to be included in a network model. For example, in the model the duration of the activity indexed 3 (line 18) could have been calculated as a complicated FORTRAN function and specified here as a user function call instead of constant. The SLAM II system has many good features as well as room for improvement. Several of the more important capabilities and some of the unique features will be discussed here.

1. Interaction with FORTRAN

It is possible to call and execute a FORTRAN subroutine from within the SLAM II network. The method is to code a special FORTRAN subroutine named EVENT with a single integer parameter. The EVENT subroutine uses the parameter as an index to determine the appropriate FORTRAN subroutine to execute. Within the SLAM II network, when an entity enters an EVENT node, the subroutine EVENT is called, which in turn calls the appropriate subroutine. The EVENT node has the following syntax:

```
EVENT, JEVNT, M;
```

JEVNT is the index to the subroutine to be called by the EVENT subroutine. M has the same meaning as in other nodes described earlier.

It is possible to send an entity into the network from a FORTRAN routine. Within the network, where the entity is to be entered from a FORTRAN program, the ENTER node should be coded. It has the following syntax:

```
ENTER, NUM, M;
```

NUM is a unique number to identify this ENTER node; M is as before. In the

FORTRAN routine, to enter an entity into the network requires the following subroutine call:

CALL ENTER (NUM, A)

A is the array for the attributes of the entity. NUM is the identification of the corresponding ENTER node in the network.

It is possible to change network status from a FORTRAN routine. Some examples are given below:

To free a resource: CALL FREE (IR, N)

N units of resource number IR are released. The resources are numbered according to the order they appear in the network declaration.

To alter a resource capability: CALL ALTER (IR, N)

The capacity of resource IR is reset to N.

To open a gate: CALL OPEN (n)

The n^{th} gate is opened. The gates are ordered according to their sequential position in the network coding.

To close a gate: CALL CLOSE (n)

The n^{th} gate is closed.

To stop an activity: The corresponding activity duration must be specially declared in the network as in: ACT, STOPA, (NTC);

In the FORTRAN routine: CALL STOPA (NTC) where NTC may be a constant, a variable, or an attribute.

It is also possible to determine an activity duration through a FORTRAN function or to assign an attribute value through a FORTRAN function. As for functions, a special FORTRAN function named USERF with one integer parameter must be included in the code. This function uses the parameter as an index to identify the function to be called. When USERF (n) is referenced in appropriate places in the network, the function USERF is called which in turn calls the function to obtain the value. Examples are given in the following.

```
ASSIGN, ATRIB (4) = USERF (1);
```

Assigns the value of the first function to attribute 4.

```
ACT, USERF (2);
```

Uses the value of the second function for duration.

The ability to include user written FORTRAN code gives added flexibility to the SLAM II network simulation.

2. Branching

Entity's flow in the network typically need to branch nonsequentially. SLAM II provides many types of branching capabilities.

Unconditional branching:

```
ACT,,,LAB1.
```

Branches to the node with label LAB1.

Conditional branching:

```
ACT,, ATRIB (2). EQ. 1, LAB2;
```

Branches to LAB2 if Attribute 2 is 1.

Probablistic branching:

```
ACT,, .25, LAB3;
ACT,, .65, LAB4;
ACT,, .10, LAB5;
```

The probabilities should add up to 1.

Delayed branching:

```
ACT, UNFRM (10, 20),, LAB5;
```

Branch with a delay or duration, according to uniform distribution.

Multiple branching:

```
AWAIT (3), PAD/1,2;
  ACT/3, 1,, FREM;
  ACT/4, X (1),, WATWN;
```

As in lines 17, 18, and 19 of the previous model.

Multiple branching produces multiple copies of the same entity. This is a useful facility as illustrated in the previous model.

One problem with multiple branching is that in situations as in the following example, the last activity becomes undefined. In the current implementation, there is not any warning given.

```
AWAIT (3), PAD/1, 2;
  ACT/3, 1,, FREM;
  ACT/4, XX (1),, WATWN;
  ACT/8, 2,, LA4;
```

<u>Combinations of above</u>:

The previous types of branching can be combined.

<u>Regular vs service activities</u>:

The key word ACT is the abbreviation of ACTIVITY. The examples given above in this subsection are all on the <u>regular activities</u>. Another type of activity is the service activity. The <u>service activity</u> is an activity emanating from a SELECT or QUEUE node. The service activities are to model single channel queues, queues with N identical servers, or multiple channel queues with non-identical servers.

An entity entering a service activity seizes one server and releases it after the given duration. Service activities also allow branching to specified nodes, but unlike regular activities, they also tie up a server for a given period. In the example simulation model, Line 11 is a service activity and all the other activities are regular activities.

3. <u>Blocking, balking, and discarding</u>

Entities trying to enter a QUEUE node or an AWAIT node may be blocked, may balk, or may be discarded. That is, a QUEUE node or an AWAIT node has a

capacity limit associated with the corresponding file. If the number of entities waiting in the file equals to the file capacity, then the new arrivals will be discarded if neither BLOCK nor BALK is specified. If BALK is specified, then the new arrival is sent to the node with the given label. If BLOCK is specified, then the entity is to be blocked in the service activity ahead of this node. Three examples are given below.

```
QUEUE (3), 0, 5, BLOCK;
```

Queue on file 3, maximum 5 entities, block specified.

```
AWAIT (2/4), RES1/1, BALK (LA1);
```

Wait for 1 resource RES1 on file 2 which has capacity 4. In case of balking, send to the node with label LA1.

```
QUEUE (2), 0, 4;
```

Queue on file 2, maximum 4 entities, excess arrivals discarded.

These features are very convenient for many situations to be modeled. However, the blocking works only if the preceding activity is a service activity. In this case, the preceding service activity is blocked when the blocking occurs. However, entities may unintentionally be lost if an AWAIT node or QUEUE node is blocked following anything other than a service activity. Such cases are not flagged by SLAM II. In the following two simulations shown in Figure 6, each generates 5 entities, but only 2 of them reach the final TERMINATE node. Three entities are lost at the AWAIT node and the QUEUE node, respectively.

Implicit discarding of entities is quite dangerous and requires very careful attention by the programmer and analyst.

4. The trace feature

The trace feature, as illustrated in the hypothetical model, is very helpful for debugging and for visualizing a dynamic phenomenon. One can specify the clock times for the start and stop of a trace. Thus, the exact operations

```
GEN,SHEN & MORRIS,TEST2 SSS6,6/23/82,1;
LIMIT,2,2,10;
NETWORK;
      CREATE,1,0,1,5;                ONE ENTITY PER TIME UNIT,MAX 5,MARK 1
      ACT/1,1;                       DURATION 1 TIME UNIT
      QUEUE(1),0,1,BLOCK;            QUEUE ON FILE 1,CAP 1,BLOCK
      ACT(1)/2,6;                    CAP 1, DURATION 6 TIME UNITS
      TERMINATE;
      ENDNETWORK;
INITIALIZE,0,50;
MONTR,TRACE,0,50,1;
FIN;
```

```
GEN,SHEN & MORRIS,TEST1 SSS5,6/23/82,1;
LIMITS,2,2,10;
NETWORK;
      RESOURCE/XRES(1),1;            ONE XRES,WAIT IN FILE 1
      CREATE,1,0,1,5;                ONE ENTITY PER TIME UNIT,MAX 5,MARK 1
      ACT/1,1;                       DURATION 1 TIME UNIT
      AWAIT(1/1),XRES/1,BLOCK;       1 XRES,FILE 1 CAP 1,BLOCK
      ACT/2,6;                       DURATION 6 TIME UNITS
      FREE,XRES/1;                   RELEASE 1 XRES
      TERMINATE;
      ENDNETWORK;
INITIALIZE,0,50;
MONTR,TRACE,0,50,1;
FIN;
```

Figure 6 - Examples of simulation in which entities are lost.

during specific periods can be observed. One deficiency about the trace is that it does not give the SELECT nodes nor the service activities. Hence, the entire picture cannot always be seen. There is also a limit of up to 5 attributes that may be displayed in the trace report.

5. The summary reports

The summary reports are generated with practically no programmer efforts. They provide quite sufficient information for most simulation purposes. Additional reports may be generated in FORTRAN subroutines called from the network if desirable.

Intermediate summary reports may be generated if they are requested on the MONTR statement. The intermediate reports, generated at specific clock times as requested, are for the status immediately before the specific clock time, not at

the end. In other words, the report for clock time 100 is for what happened immediately before clock time 100. This may be misleading at times.

A problem with the intermediate reports which may cause confusion is that an activity that has not ended is not counted. Consider an activity which is supposed to end for an entity at the beginning of clock time 5. The actual processing of its ending is at the beginning of clock time 5. The intermediate report for clock time 5 will not include this entity in the activity's entity count even if the entity entered the activity at, say clock time 2. This result is confusing and misleading. A person reading several intermediate reports may find that sometimes the correct number of entities are in the entity counts, but at other times the entity counts are a little short.

However, in the final summary report, this problem is corrected. In other words, if the final summary report is also for clock time 5, then the problem described above disappears from the final summary report.

6. The symbolism

The nodes and the branches of the network may be represented symbolically as illustrated earlier. The network should be drawn graphically to help visualize the model. This helps in detecting errors and in building a good model. It amounts to flowcharting for other types of computer programming.

The symbolism used by the Pritsker and Associates, Inc. is complicated. The graphic symbols are based on those used in Q-GERT [6], but they are difficult to learn and hard to retain. A simple circle or a rectangle is sufficient to represent any node. A key word may be added to the circle or rectangle to indicate the type of node it represents. Such simplified notation would cause no problems for the experienced users of SLAM II or Q-GERT, but this would definitely help those new to the system or those who do not work with it very often.

Comments in English written beside the nodes in the network graph also proved helpful. This idea was used in our hypothetical model. Without such comments, the network graph would be difficult to understand.

The parameters of the nodes and the activities are position-significant. Therefore, use of special symbols for the parameters are not necessary. For example, consider the following two node statement syntax representations:

CREATE, TBC, TF, MA, MC, M;

PREEMPT (IFL)/PR, RLBL, SNLBL, NATR, M;

From a human engineering point of view, a simplified representation such as

CREATE, A, B, C, D, E;

PREEMPT, A, B, C, D, E, F;

is easier to understand and less error prone for programmers to verify their codings. Because in this simplified representation, a programmer can more easily detect, for instance, the misposition of a parameter, or the missing of a parameter. Such representation is used in many computer languages including GPSS.

7. System limitations

Any computer software has limitations on data configurations. A simulation language processor is not immune to this problem. However, a good system should allow sufficiently high limitations which hopefully will not be reached under most situations. Modern computer programming languages such as PASCAL [7], SIMULA [5], and many others allow users to dynamically declare their space requirements.

SLAM II does allow the users to dynamically declare certain space requirements, but they are subject to rather stringent constraints. The constraints are listed in the following.

(1) The number of attributes of each entity ≤ 98.

The same amount of space is reserved for the purpose of attributes for all the different entity types. If most entity types have only one or two attributes but one entity type has, say, 90 attributes, then the space for 90 attributes is reserved for all the entities. In such a case, space may be greatly wasted.

(2) The number of entities that can concurrently exist in the model $\leq$ an indefinite number.

This number varies according to the size of the model. In the previous hypothetical model, a declaration of 791 causes input error. For a complicated and large model with large attribute size, this number may be down to 100 or less.

(3) The number of files < 100.

This number is a limit on the number of QUEUE and AWAIT nodes that may exist in the model. In case some nodes require more than one file each, then less than 99 such nodes are allowed.

(4) The number of indexed activities $\leq$ 50.

This is not a limit on the number of activities allowed in a model. However, only the indexed ones have statistics maintained and reported. On the other hand, service activities do have utility statistics reported whether they are indexed or not.

All the above restrictions may be modified if a user knows how to appropriately change the source SLAM II programs. However, requiring users to modify the source code is not a good practice. It means another level of complexity and possibly user introduced errors. At the same time, most of these limits are set high enough so that many users would not be required to change the limits.

8. <u>Model integrity</u>

A good programming language processor should help programmers write correct programs. Error checkings should be beyond the syntax within single

lines or statements. SLAM II does perform such checkings and provides many helpful error messages. The following gives some examples on what SLAM II does and does not do.

Entities are placed on specific files to wait in queues at QUEULE nodes or to wait for resources at AWAIT nodes. Such files should be unique for each QUEUE node or AWAIT node. When a QUEUE node and an AWAIT node both use the same file, a SLAM input error is detected before the simulation is attempted. An input error is also detected if two QUEUE nodes use the same file. However, if two AWAIT nodes use the same file, then no error is detected. When an AWAIT node uses a file which is not specified for the corresponding resource, no error is detected.

Two different resources should have mutually exclusive sets of files associated with them for the entities waiting for the resources. SLAM II does not detect the error if this restriction is violated. Any undetected error concerning the files for AWAIT, QUEUE, and RESOURCE may cause erroneous entity flow in the network and possibly unintentional discarding of some entities.

The possibility of discarding entities without explicitly terminating them may cause accidental loss of entities in various parts of a network model. Another example in which this may happen is the misuse of BLOCK as discussed previously.

Another point concerning integrity that was discussed earlier is that the intermediate summary reports do not typically give correct entity counts.

The summary reports use the term "utilization" in an unconventioanl manner, which may cause a misinterpretation of the result. The term is generally understood to be the time used divided by time available; thus, it is always less than or equal to one. In the summary report, many utilization values are

greater than one. It appears that "number of entities" may be a more appropriate term than "utilization" in the reports.

9. Some Other factors

The availability and user population are often a factor in choosing a computer software system. SLAM II currently is available on a large number of different computers. According to the company, its user installations numbered about 400, including some duplicate installations in single institutions. Among them about 200 are in universities. Thus, the availability and portability are reasonably good. Its user population is small but growing rapidly.

SLAM II documentation has room for improvement. The textbook entitled Introduction to Simulation and SLAM by Pritsker and Pegden [1] contains the most information on the language, but is not up to date. A separate manual entitled SLAM II, Enchanced Simulation Capabilities [2] contains the updated information. The text book does have many examples which provide helpful illustrations of the use of the different features of the language. The SLAM II source listings have less than adequate remarks to explain the different parts of the programs.

Special training courses are offered by Pritsker and Associates, Inc., however, such training courses may be too brief for people without prior experience in a comparable simulation language.

The company does provide very good user services, and a telephone line is maintained to answer questions or to help programmers.

Experiments to test or to compare the SLAM II execution efficiency have not been made. As computers get cheaper rapidly, the execution efficiency is often becoming a secondary factor to be considered nowadays.

IV. Conclusion

SLAM II is a general purpose simulation language that offers the user a language system which in a relatively short period of time can be used to develop network simulation models that can be successfully run and provide useful results in a standardized output. The flexibility of the system should allow discrete event and continuous modeling in combination with the network simulation. The use of the language for more extensive modeling applications will require more care on the part of the user because the system also has limitations which may cause undetected loss of entities, duplicate use of the same file, and reports which may mislead the user. New users of this language system need to be especially cautioned about these model characteristics so that these problems may be avoided.

References

1. Pritsker, A.A.B. and C. D. Pedgen, Introduction to Simulation and SLAM. Systems Publishing Corp. and Holsted Press, N.Y. 1979.

2. Duket, S. D., J. J. O'Reilly, and R. J. Hannan, SLAM II Enhanced Simulation Capabilities, Pritsker & Associates, Inc., June 1981.

3. Schriber, T., Simulation Using GPSS, John Wiley, 1974.

4. Kiviat, P. J., R. Villanueva, and H. Markowitz, SIMSCRIPT II.5 Programming Language, Consolidated Analysis Centers, Inc., 1973.

5. Birtwhistle, G. M., O. Dahl, B. Myhrhang, and K. Nygaard, SIMULA Begin, Auerbach, 1973.

6. Pritsker, A.A.B., Modeling and Analysis Using Q-GERT Networks, Halsted Press and Pritskerand Associates, Inc. 1977.

7. Richards, J. L., PASCAL, Academic Press, 1982.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-84605		

4. Title and Subtitle

NETWORK SIMULATION USING THE SIMULATION LANGUAGE FOR ALTERNATE MODELING (SLAM II)

5. Report Date

February 1983

6. Performing Organization Code

506-51-13-04

7. Author(s)

Stewart Shen* and W. Douglas Morris

8. Performing Organization Report No.

10. Work Unit No.

9. Performing Organization Name and Address

NASA Langley Research Center
Hampton, VA 23665

11. Contract or Grant No.

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, DC 20546

13. Type of Report and Period Covered

Technical Memorandum

14. Sponsoring Agency Code

15. Supplementary Notes

*Old Dominion University
Norfolk, VA 23508

16. Abstract

Computer simulation is a widely used technique for problem solving, system analysis, and design. Many specialized languages have been developed as an aid to implementing simulation studies. The simulation language for alternate modeling (SLAM II) is a general purpose language that combines network, discrete event, and continuous modeling capabilities in a single language system. The efficacy of the system's network modeling is examined and discussed. Examples are given of the symbolism that is used, and an example problem and model are derived. The results are discussed in terms of the ease of programming, special features, and system limitations. The system offers many features which allow rapid model development and provides an informative standardized output. The system also has limitations which may cause undetected errors and misleading reports unless the user is aware of these programming characteristics.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Computer, Simulation Modeling, Network, Simulation Language, Discrete Event Simulation	Unclassified - Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	32	A03

* For sale by the National Technical Information Service, Springfield, Virginia 22161